CS10720 Problems and Solutions

Thomas Jansen

Today: More Representations for Integers

February 1st

Announcements

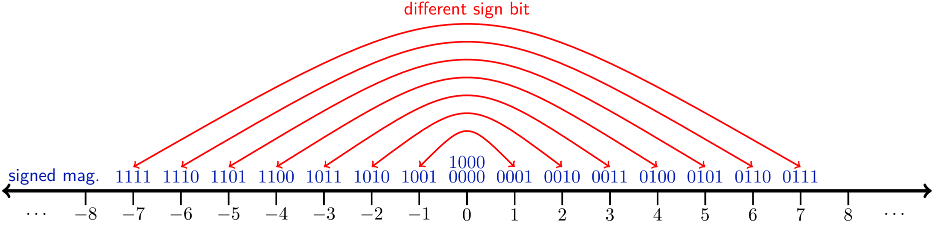
Portfolio feedback sent out today
Contact me if you have not received it
or of you think something is not right.

Contact me if your portfolio feedback asks you to contact me
because I have an important question.

Revised deadline for all portfolio submissions
each Friday, 7pm

Plans for Today

Recap: Signed Magnitude Representation



Summary of facts about signed magnitude representation

- leftmost bit acts as sign bit
- non-negative numbers: standard binary encoding
  with sign bit 0
- negative numbers: standard binary encoding
  with sign bit 1
- 0 has two representations, 000000 and 100000
- largest number represented by $0111\cdots11$: $2^{l-1}-1$ (l bits)
- smallest number represented by $1111\cdots11$: $-2^{l-1}+1$ (l bits)

Integers in Binary Representation: One's Complement

Remember signed magnitude representation
sign bit 0 represents non-negative number
sign bit 1 represents negative number
other bits in standard binary encoding

Simple idea use the same idea for the sign bit
but flip all bits for negative numbers

Example $(11)_{10}$ represented as 001011
$(-11)_{10}$ represented as 110100

How is 0 represented?

Observation 000000 represents $(-1)^0 \cdot 0 = 1 \cdot 0 = 0$
111111 represents $(-1)^1 \cdot 0 = -1 \cdot 0 = 0$

Remark two different representations for 0 are unpleasant
because '$+0 = -0$' but $000000 \neq 111111$ on bit-level
makes comparison harder for computers to perform

Example: Conversion Decimal $\rightarrow$ One's Complement

Example Convert -19 to One's Complement Rep. with 6 bits

Conversion of 19 into binary
(here with repeated division, as before)
$19/2 = 9$ R 1 thus, least significant bit is 1
$9/2 = 4$ R 1 thus, next bit is 1
$4/2 = 2$ R 0 thus, next bit is 0
$2/2 = 1$ R 0 thus, next bit is 0
$1/2 = 0$ R 1 thus, next bit is 1
since we are at 0 all remaining bits are 0
Result 10011

Observation -19 is negative, so invert all bits

Result 101100 (because we use 6 bits as length)

Example: One's Complement → Decimal

Example Convert 100110 from One's Complement Rep.

Observation first bit is 1, thus number is negative
thus, invert all bits

Conversion 11001 into decimal
$(11001)_2$
$= 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4$
$= 1 + 8 + 16 = 25$
Result 25

Result -25

The Extreme Values

What is the largest number you can represent with $l = 6$ bits in one's complement representation?

Representation 011111

Value $1+2+4+8+16=31$

And in general, for arbitrary l?

Representation $\underbrace{0111\cdots11}_{0 \text{ and } l-1 \text{ 1-bits}}$

Value $1+2+4+8+\cdots+2^{l-2}=2^{l-1}-1$

What is the smallest number you can represent with $l = 6$ bits in signed magnitude representation?

Representation 100000

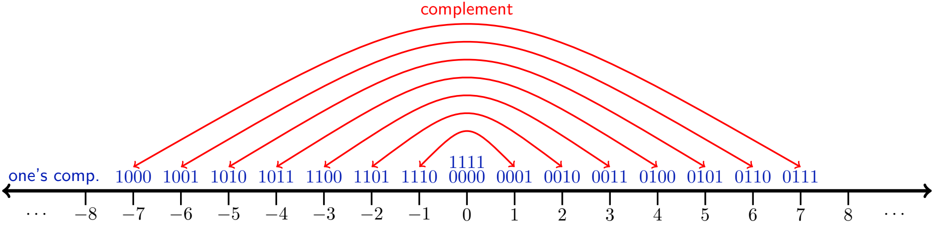
Value $-(1+2+4+8+16)=-31$

And in general, for arbitrary l?

Representation $\underbrace{1000\cdots00}_{1 \text{ and } l-1 \text{ 0-bits}}$

Value $-(1+2+4+8+\cdots+2^{l-2}) = -\left(2^{l-1}-1\right) = -2^{l-1}+1$

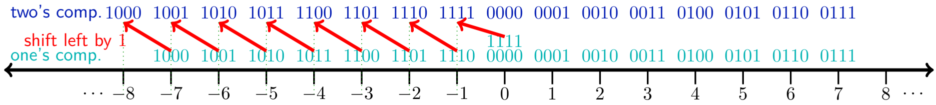
Summary: One's Complement Representation



Summary of facts about one's complement representation

- leftmost bit acts as sign bit
- non-negative numbers: standard binary encoding
  with sign bit 0
- negative numbers: complement of standard binary encoding
- 0 has two representations, 000000 and 111111
- largest number represented by $0111\cdots11$: $2^{l-1}-1$ (l bits)
- smallest number represented by $1000\cdots00$: $-2^{l-1}+1$ (l bits)

Towards Two's Complement Representation



Observation both representations wasteful
using two different bit patterns for the same number, 0
(and introducing additional problems by doing this)

Idea modify one's complement representation
using only 0000 to represent 0
and do something useful with the free bit pattern 1111

Example: Conversion Decimal $\rightarrow$ Two's Complement

Example Convert -19 to Two's Complement Rep. with 6 bits

Conversion of 19 into binary
(here with repeated division, as before)
$19/2 = 9$ R 1 thus, least significant bit is 1
$9/2 = 4$ R 1 thus, next bit is 1
$4/2 = 2$ R 0 thus, next bit is 0
$2/2 = 1$ R 0 thus, next bit is 0
$1/2 = 0$ R 1 thus, next bit is 1
since we are at 0 all remaining bits are 0
Result 10011

Observation -19 is negative, so invert all bits and add 1
$101100 + 1 = 101101$ (because we use 6 bits as length)

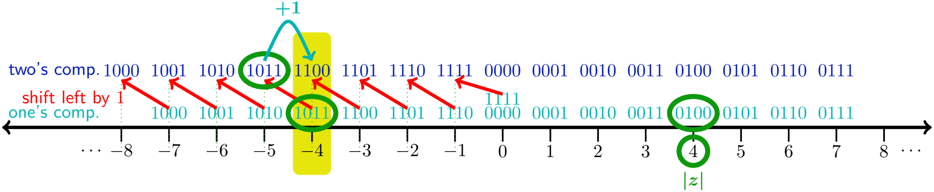
Result 101101 (because we use 6 bits as length)
Why does 'invert all bits and add 1' work?

Method for Converting Decimal ⇝ Two's Complement

Method

1. Convert absolute value into standard binary encoding.
2. If number negative, invert all bits and add 1.

Why does this work?



Consider standard binary encoding of $|z|$
Its inverse needs to be shifted 1 to the right
to correct for shift to left by 1; corresponding to adding 1.

Example: Two's Complement $\rightarrow$ Decimal

Example Convert 100110 from Two's Complement Rep.

Observation first bit is 1, thus number is negative
thus, invert all bits and add 1

Conversion $11001 + 1 = 11010$ into decimal
$(11010)_2$
$= 0 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 + 1 \cdot 2^4$
$= 2 + 8 + 16 = 25$
Result 25

Result -25

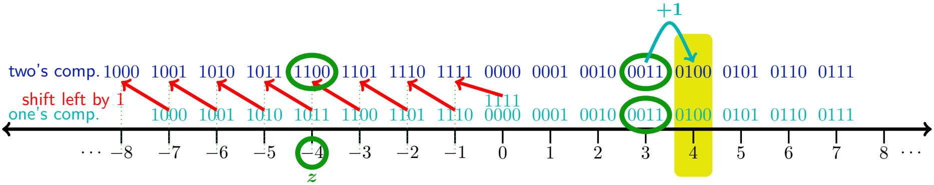
Why does 'invert all bits and add 1' work in this direction, too?

Method for Converting Two's Complement $\rightsquigarrow$ Decimal

Method

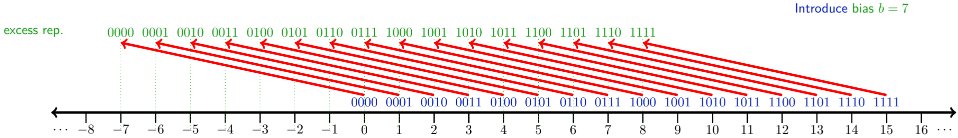
1. If first bit is 1, invert all bits and add 1.
2. Translate from standard binary encoding.

Why does this work?



Consider two's complement encoding of z
Its inverse needs to be shifted 1 to the right
to correct for shift to left by 1; corresponding to adding 1.

Excess Representation



Summary of facts about excess representation

- works with any bias $b \in \mathbb{Z}$
- bias $b = 2^{l-1} - 1$ most common (e.g. $b = 7$ for $l = 4$, $b = 31$ for $l = 6$)
- with $b = 2^{l-1}$ leftmost bit almost a sign bit
- 0 has at most one representation, b
- largest number represented by $1111\cdots11$: $2^l - 1 - b$
- smallest number represented by $0000\cdots00$: $-b$
- order of numbers equal to standard binary encoding

Example: Conversion Decimal $\rightarrow$ Excess Representation

Example Convert -19 to Excess Rep., 6 bits, excess $2^5 - 1 = 31$

Observation we need to represent $-19 + 31 = 12$

Conversion of 12 into binary
(different methods available, here using repeated division)
$12/2 = 6$ R 0 thus, least significant bit is 0
$6/2 = 3$ R 0 thus, next bit is 0
$3/2 = 13$ R 1 thus, next bit is 1
$1/2 = 0$ R 1 thus, next bit is 1
since we are at 0 all remaining bits are 0
Result 1100

Result 001100

Example: Excess Representation $\rightarrow$ Decimal

Example Convert 100110 from Excess Rep. with excess 31

Conversion 100110 into decimal

$$(100110)_2$$
$$= 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 + 0 \cdot 2^4 + 1 \cdot 2^5$$
$$= 2 + 4 + 32 = 38$$

Result 38

Remember in excess representation with excess 31
38 represents $38 - 31 = 7$

Result 7

http://onlineted.com

Different Representations: Why?

Why do we consider different binary representations for integers?
Why do we not simply consider only the best?

Insight what is the best binary representation for integers may be depend on what we want to do with it

What do we want to do with integers in our computer?

- store them
- add or subtract them
- perform even more complex operations (like multiplication, division, square roots, . . .)
- change their sign
- compare them

How do comparisons work in different representations?

Comparing Integers in Signed Magnitude Rep. (Part 1)

Problem compare a with b http://onlineted.com
(both in signed magnitude representation with $l = 6$ bits)

Case 1 different first bits

- a starts with 1, b starts with 0 (not both representing 0)
  $a < b$ (example $100101 < 000001$)
- a starts with 0, b starts with 1 (not both representing 0)
  $a > b$ (example $000011 > 101101$)
- both representing 0
  $a = b$ (example $100000 = 000000$)

Case 2 both first bits 0

- at leftmost differing position a is 1, b is 0
  $a > b$ (example $010101 > 010001$)
- at leftmost differing position a is 0, b is 1
  $a < b$ (example $001011 < 001101$)
- all bits equal
  $a = b$ (example $101001 = 101001$)

Comparing Integers in Signed Magnitude Rep. (Part 2)

Problem compare a with b
(both in signed magnitude representation with $l = 6$ bits)

Case 2 both first bits 0

- at leftmost differing position a is 1, b is 0
  $a > b$ (example $010101 > 010001$)
- at leftmost differing position a is 0, b is 1
  $a < b$ (example $001011 < 001101$)
- all bits equal
  $a = b$ (example $001001 = 001001$)

Case 3 both first bits 1

- at leftmost differing position a is 1, b is 0
  $a < b$ (example $110101 < 110001$)
- at leftmost differing position a is 0, b is 1
  $a > b$ (example $101011 > 101101$)
- all bits equal
  $a = b$ (example $101001 = 101001$)

Comparing Integers in One's Complement Rep. (Part 1)

Problem compare a with b http://onlineted.com
(both in one's complement representation with $l = 6$ bits)

Case 1 different first bits

- a starts with 1, b starts with 0 (not both representing 0)
  $a < b$ (example $100101 < 000001$)
- a starts with 0, b starts with 1 (not both representing 0)
  $a > b$ (example $000011 > 101101$)
- both representing 0
  $a = b$ (example $111111 = 000000$)

Case 2 both first bits 0

- at leftmost differing position a is 1, b is 0
  $a > b$ (example $010101 > 010001$)
- at leftmost differing position a is 0, b is 1
  $a < b$ (example $001011 < 001101$)
- all bits equal
  $a = b$ (example $101001 = 101001$)

Comparing Integers in One's Complement Rep. (Part 2)

Problem compare a with b
(both in signed magnitude representation with $l = 6$ bits)

Case 2 both first bits 0

- at leftmost differing position a is 1, b is 0
  $a > b$ (example $010101 > 010001$)
- at leftmost differing position a is 0, b is 1
  $a < b$ (example $001011 < 001101$)
- all bits equal
  $a = b$ (example $001001 = 001001$)

Case 3 both first bits 1

- at leftmost differing position a is 1, b is 0
  $a > b$ (example $110101 > 110001$)
- at leftmost differing position a is 0, b is 1
  $a < b$ (example $101011 < 101101$)
- all bits equal
  $a = b$ (example $101001 = 101001$)

Comparing Integers Represented in Binary

Case 1: Non-Negative Numbers

Observe signed magnitude, one's complement, two's complement all identical

Remember $0001\,0111 = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + \cdot 2^4$

Observe $\forall p \in \mathbb{N}\colon 2^p > \sum_{i=0}^{p-1} 2^i$ (since $\sum_{i=0}^{p-1} 2^i = 2^p - 1$)

Consequence left-most differing bit decides

Example $0010110 > 0001111$

What about excess representation?

Observation also works like this
because $a > c$ if and only if $a + b > c + b$
(so it does not hurt to compare $a + b$ with $c + b$ instead of a and c)

Comparing Integers Represented in Binary

Case 2: Negative Numbers

Signed magnitude requires reversal of logic
because $a > c$ implies $-a < -c$
but sign change in signed magnitude representation
changes leading 0-bit to 1-bit
leaving comparison result unchanged

One's complement works as before
because $a > c$ implies $-a < -c$
and inverting all bits inverts comparison result

Two's complement works as before
because $a > c$ implies $-a < -c$
and inverting all bits inverts comparison result
and adding 1 does not change this

Excess works as before
because $a > c$ if and only if $a + b > c + b$
and that holds for positive and negative a and c

Comparing Integers Represented in Binary

Case 3: Negative and Positive Numbers

Signed magnitude special case
because result depends only on sign bit

One's complement special case
because result depends only on left-most bit

Two's complement special case
because result depends only on left-most bit

Excess works as before
because $a > c$ if and only if $a + b > c + b$
and that holds for positive and negative a and c

Summary only excess representation works the same in all cases
all other representations require case distinctions

Comparing Representations for Integers (Using l Bits)

Property	**Signed M.**	**One's C.**	**Two's C.**	**Excess** $\boldsymbol{b = 2^{l-1} - 1}$
Rep. of 0	$000\cdots00$ or $100\cdots00$	$000\cdots00$ or $111\cdots11$	$000\cdots00$	$011\cdots11$
Rep. of 1	$000\cdots01$	$000\cdots01$	$000\cdots01$	$100\cdots00$
Rep. of -1	$100\cdots01$	$111\cdots10$	$111\cdots11$	$011\cdots10$
Largest rep.	$011\cdots11$	$011\cdots11$	$011\cdots11$	$111\cdots11$
Smallest rep.	$111\cdots11$	$100\cdots00$	$100\cdots00$	$000\cdots00$
Largest num.	$2^{l-1} - 1$	$2^{l-1} - 1$	$2^{l-1} - 1$	2^{l-1}
Smallest num.	$-\left(2^{l-1} - 1\right)$	$-\left(2^{l-1} - 1\right)$	-2^{l-1}	$-\left(2^{l-1} - 1\right)$

Summary Properties of Different Representations

Representation	sign change	comparisons	additions
signed magnitude	very easy	hard	hard
one's complement	easy	hard	hard
two's complement	not so easy	hard	easy
excess	very hard	easy	very hard

- if comparisons are most important
  then use excess representation
- if additions are most important
  then use two's complement representation

Remember   it depends on the application
what the best representation is

Summary & Take Home Message

Things to remember

- representation of integers: one's complement, two's complement, excess
- comparing representations of integers
  - signed magnitude: the same as for decimals; sign change very easy; easy to read
  - one's complement: sign change easy
  - two's complement: unique representation for each number; range of numbers larger by 1; (not seen) supports addition
  - excess: unique representation for each number; range of numbers larger by 1; supports comparisons

Take Home Message

- Knowing the basics is important.
- Numbers can be represented in different formats.
- There is no unique best format. It depends on the application.
- Standards are important.

Lecture feedback `http://onlineted.com`